UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE ECONOMIA
MONOGRAFIA DE BACHARELADO

TRABALHANDO DE GRAÇA? ANÁLISE DO COMPORTAMENTO DOS AGENTES ECONÔMICOS PRODUTORES DE SOFTWARE LIVRE

RODRIGO TASSINARI DE OLIVEIRA
matrícula nº 101113556

ORIENTADOR: Prof. Paulo Bastos Tigre

SETEMBRO 2010

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE ECONOMIA
MONOGRAFIA DE BACHARELADO

TRABALHANDO DE GRAÇA? ANÁLISE DO COMPORTAMENTO DOS AGENTES ECONÔMICOS PRODUTORES DE SOFTWARE LIVRE

RODRIGO TASSINARI DE OLIVEIRA
matrícula nº 101113556

ORIENTADOR: Prof. Paulo Bastos Tigre

SETEMBRO 2010

As opiniões expressas neste trabalho são de exclusiva responsabilidade do autor.

Dedico este trabalho a Richard M. Stallman, por em sua busca pela liberdade do software, mudar o mundo.

AGRADECIMENTOS

Agradeço a meu pai por encaminhar meus primeiros passos digitais quando ainda criança, e a minha mãe pela compreensão e paciência ao longo dos anos. Muito agradecido estou também a professora Maria Silvia Possas, por me fazer persistir, e a meu orientador, professor Paulo Tigre, pelo incentivo e disponibilidade. Por fim, agradeço a Carol, pelo apoio e por acreditar em mim mais do que eu mesmo. Obrigado a todos.

RESUMO

Este trabalho analisa o comportamento dos indivíduos e das empresas competitivas produtores de software livre, procurando entender os incentivos que levam estes agentes à investirem seus recursos escassos na produção do bens essencialmente públicos sem nenhuma recompensa pecuniária direta, na grande maioria dos casos.

ÍNDICE

INTRODUÇÃO

Todos os dias, milhares de pessoas altamente qualificadas empregam várias horas do seu tempo e seu esforço mental à realização de trabalhos complexos, pelos quais elas não serão remuneradas. Além disso, o produto deste trabalho será distribuído gratuitamente e de maneira tal que qualquer pessoa no planeta poderá dele usufruir, sem nenhum controle por parte dos criadores originais do produto.

O modelo de produção e consumo descrito no parágrafo anterior parece estranho, pois contradiz princípios econômicos básicos, além do senso comum. Por quê alguém trabalharia de graça? No entanto, este modelo ocorre diariamente há vários anos no desenvolvimento de software-livre. Programadores de todas as partes do mundo decidem voluntariamente trabalhar em projetos de software cuja licença de uso permite que qualquer pessoa use, redistribua e modifique o sistema, com total acesso ao código-fonte do mesmo, efetivamente forçando tal software a ser distribuído gratuitamente.

Esta monografia objetiva entender os motivos que levam estes programadores a participarem ativamente no desenvolvimento de software-livre, em geral sem nenhuma remuneração pecuniária pelos seus esforços.

A monografia está organizada em quatro capítulos. No primeiro capítulo é feito uma análise das características econômicas de software em geral (livre ou não) enquanto um bem de informação. No capítulo seguinte, é definido o que caracteriza um software como “livre”, e como isso o diferencia do modelo tradicional de software proprietário. Um resumo da origem e história do software-livre no mundo é brevemente apresentado ao fim do capítulo.

O terceiro capítulo analisa a fundo a produção de software no modelo livre. Após uma breve descrição sobre a estrutura padrão de um projeto de software-livre e sua origem, são investigados os motivos que levam os agentes econômicos a contribuírem com tais projetos. A motivação dos indivíduos é tratada primeiramente, analisando os possíveis incentivos intrínsecos e extrínsecos atuantes sobre os agentes, segundo a literatura pesquisada. Em seguida o foco estende-se às empresas competitivas, visando entender os possíveis motivos para participarem do desenvolvimento de um produto disponível gratuitamente a qualquer

interessado.

No quarto capítulo é feito um estudo de caso sobre uma microempresa usuária e produtora de software-livre. Através de entrevista com um dos sócios da empresa, avaliou-se o papel que o software-livre possui nas atividades da empresa, além das motivações da mesma e de seus sócios e funcionários a produzirem software-livre. Em seguida compara-se a realidade particular desta empresa e sua relação com software-livre ao material previamente estudado nos capítulos anteriores, situando as observações coletadas via entrevista na teoria apresentada.

Por fim, a monografia apresenta uma conclusão, onde é feito uma retomada da discussão apresentada. A miríade de possíveis incentivos atuantes sobre os agentes econômicos quando da produção de software-livre é revista. Em seguida, argumenta-se que em um projeto de software-livre típico, a heterogeneidade dos agentes econômicos, no tocante aos incentivos individuais à participação, permite que diversos incentivos atuem ao mesmo tempo, explicando o improvável sucesso do projeto.

Capítulo I – Principais conceitos

I.1 – O software como informação

Um *software*, ou programa de computador, nada mais é do que uma série de rotinas e comandos pré-determinados, escritos de maneira que um computador possa compreendê-los e executá-los. Shapiro e Varian (1999) caracterizam software como um bem de informação, que em sua definição corresponde a "qualquer coisa que puder ser digitalizada – codificada como um fluxo de *bits*" (p. 15). Krugman, Wells e Myatt (2005) usam uma definição mais abrangente: um bem de informação é um bem cujo valor provém da informação que ele contém, admitindo assim que bens físicos que contém informação (digitalizada ou não) também se enquadrem nesta categoria (p. 520). Em ambas as definições, porém, a produção de bens de informação possui como característica particular altos custos fixos e baixos custos marginais de produção, estrutura esta encontrada nos *softwares* e também na produção de livros, filmes, músicas, notícias, etc..

A distribuição de um *software* pode ser feita, de maneira geral, de duas maneiras: em formato de código-fonte ou em código-objeto (também chamado de formato binário). Código-fonte é o *software* escrito em linguagens específicas de programação, como Java, C e Python, compreensíveis por qualquer pessoa com conhecimento destas linguagens. É a "receita" do *software*, e expõe todo o seu funcionamento à quem tiver acesso a este formato. Código-objeto, por sua vez, é o código-fonte compilado, isto é, transformado em seqüências de zeros e uns (e daí o nome de código binário) compreensíveis para os processadores de um computador, porém de pouca utilidade para uma pessoa. A maioria dos *softwares* são distribuídos apenas em sua forma binária, enquanto seu código-fonte correspondente costuma ser guardado pelas empresas como segredo industrial, protegido por leis de propriedade intelectual e até por patentes.

Apesar de ser extremamente difícil obter compreensão sobre o modo de funcionamento e sobre as rotinas internas de um *software* a partir do código binário apenas, esta possibilidade existe. Para contornar isso, porém, as empresas produtoras de *softwares* costumam fornecer o código binário de seus produtos juntamente com uma licença de uso, que deve ser aceita pelo usuário para que o mesmo possa utilizar o *software*. Estas licenças costumam impor regras estritas sobre o que constitui uso legal do produto e também costumam proibir tentativas de aplicação de engenharia reversa sobre o código binário do mesmo.

I.2 - Produção de bens digitais

Informação, como mencionado acima, pode ser entendida como tudo aquilo que pode ser digitalizado, e cujo valor depende da informação contida ao invés do custo marginal de produção. O conteúdo de jornais e revistas, livros, programas de televisão, tudo isto são bens de informação. No entanto, como observam Shapiro e Varian (1999), a estrutura de custos da produção de informações é bastante incomum. Em suas palavras:

> *A informação é cara de* produzir*, mas barata para* reproduzir. *Livros que custam centenas de milhares de dólares para produzir podem ser impressos e encadernados por um ou dois dólares, e filmes de US$ 100 milhões podem ser copiados em fita de vídeo por alguns centavos. (p. 15 – 16, grifo no original)*

Em termos econômicos, a produção de bens de informação possui altos custos fixos, porém baixos custos marginais. Os custos de produção da informação estão todos concentrados na produção da primeira cópia da mesma, sendo mínimos os custos de produção de cópias subseqüentes. Esta estrutura de custos implica em uma formação de preços de acordo com o valor que o bem possui para o consumidor, e não baseado no custo unitário do produto. Outra conseqüência é a presença inerente de uma forte economia de escala da produção, principalmente em situações onde o custo marginal da reprodução da informação é desprezível, como em um *download* da Internet.

O principal componente dos custos fixos na produção de um bem de informação são os custos amortizados (*sunk costs*), ou seja, custos que não podem ser recuperados uma vez iniciada a produção. Além disso, "os custos amortizados geralmente têm de ser pagos adiantado, antes do início da produção" (Shapiro e Varian, 1999). Os custos variáveis da informação também possuem um comportamento interessante, pois "o custo de produção de uma cópia adicional em geral não aumenta, mesmo que se faça um grande número de cópias" (Shapiro e Varian, 1999). Não costuma haver, assim, restrições de oferta por parte dos produtores de informação: pode-se produzir poucas ou muitas cópias da mesma, todas ao mesmo custo unitário.

Além das economias de escala, as empresas produtoras de bens de informação podem incorrer em economias de escopo, diluindo seus custos ao ampliar a gama de produtos e serviços oferecidos. A informação fornece oportunidades de maior aproveitamento de economias de escopo do que outros tipos de produtos pois ela pode ser empacotada de muitas formas diferentes, através do aproveitamento de uma mesma fonte. Um filme de cinema pode

ser comercializado em outros formatos além do original, como DVD, VHS e exibições em emissoras de TV abertas e fechadas, por exemplo.

Esta estrutura de custos peculiar implica também que os mercados de bens de informação não serão perfeitamente competitivos. Mesmo que não haja nenhum grande participante que se sobressaia aos demais, o preço de produtos similares será influenciado pelas estratégias das firmas do mercado. A facilidade de cópia da informação permite que a mesma seja transformada em *commodity*, caso não haja diferenciação entre os fornecedores do mercado. Ainda segundo Shapiro e Varian (1999):

> *Uma vez que várias empresas tenham amortizados os custos necessários para criar o produto [...], as forças competitivas tendem a conduzir o preço para o custo marginal, o custo de produzir uma cópia 'adicional'. (p. 39)*

Os autores identificam então haver apenas duas estruturas sustentáveis para mercados de bens de informação. O primeiro modelo seria aquele onde existe uma empresa dominante, com expressiva participação no mercado, geralmente conquistada por ter sido a empresa pioneira no mesmo. Esta empresa dominante possui grandes vantagens de custo sobre seus concorrentes, devido à seu tamanho e economia de escala. Um segundo modelo seria o de mercados de produtos diferenciados, onde temos "numerosas empresas produzindo o mesmo 'tipo' de informação, mas com muitas variedades diferentes", sendo este modelo a estrutura de mercado mais comum na realidade. Há, também, mercados onde ocorrem ambos os modelos simultaneamente.

A diversificação, ou criação de versões, dos bens de informação, é então variável chave para impedir a comoditização de um produto e permitir o aproveitamento de economias de escopo. A criação de versões diferentes de um mesmo produto permite alcançar consumidores diferentes, que atribuem valores diferentes ao mesmo produto. Uma empresa de *software* pode lançar duas versões de um aplicativo específico, uma voltada para usuários exigentes, dispostos a pagar mais pelo produto, que contem todas as funções avançadas do produto, e outra versão direcionada a usuários ocasionais, dispostos a pagar menos pelo mesmo produto, que contem várias funções desabilitadas. A empresa estará assim alcançando ambos os consumidores, e vendendo essencialmente o mesmo produto à preços diferentes, de acordo com o valor que cada consumidor dá para o produto.

Outras maneiras de diferenciação entre produtos e serviços podem ocorrer pela

qualidade do serviço, rapidez de acesso à informação ou grau de especialização da mesma. Um serviço de acompanhamento de preços de ações da Bolsa de Valores pode cobrar barato para prestar um serviço que disponibiliza os dados com atraso de 20 minutos, por exemplo, e cobrar bem mais caro àqueles consumidores que desejam obter os dados em tempo real.

I.3 - Padrões e aprisionamento tecnológico

Enquanto que uma pessoa que adquire um automóvel não se preocupa se o mesmo poderá rodar livremente nas ruas de sua cidade, ou se poderá ser abastecido em qualquer posto de gasolina, o mesmo não pode ser dito sobre um comprador de um bem de informação, como um *software* por exemplo. Este comprador deverá estar atento para a compatibilidade deste *software* com seu computador, seu sistema operacional, seus arquivos existentes, enfim, o novo *software* deve operar dentro de um padrão específico que assegure a comunicação com outros bens de informação do interesse do comprador.

Isto ocorre pois a informação normalmente é "estocada, manipulada e comunicada utilizando-se um 'sistema' que consiste em múltiplas peças de hardware e software e porque precisa-se de treinamento especializado para a utilização de sistemas específicos" (Shapiro e Varian, 1999, p. 139). Em outros termos, a estrutura da informação obedece a padrões específicos, e dois bens de informação distintos geralmente possuem padrões também diferentes de funcionamento.

Na verdade, pode-se identificar o uso de padrões nos bens tradicionais: no exemplo acima, o tamanho do automóvel em relação às ruas da cidade e a bitola das bombas de combustível podem ser considerados alguns dos padrões aos quais os carros estão sujeitos. Porém, estes padrões geralmente são abertos, comuns à toda indústria, e amplamente consolidados. O diferencial da economia da informação é a presença de muitos padrões fechados, específicos de um fornecedor ou grupo de fornecedores por exemplo.

O uso de padrões fechados, ou proprietários, é a principal fonte de custos de troca e aprisionamento tecnológico na economia da informação. Varian e Shapiro (1999) definem os custos de troca como aqueles que os agentes econômicos deverão arcar quando mudarem de um sistema de informação para outro, como a troca de uma tecnologia por outra ou até mesmo a troca de marcas ou fornecedores de um mesmo produto ou serviço. Quando estes custos de troca se tornam substanciais, os agentes econômicos enfrentam o aprisionamento, isto é, eles estarão presos à determinada tecnologia ou fornecedor.

Alguém que adquire um computador com sistema operacional Microsoft Windows, por exemplo, enfrentará custos de troca posteriormente ao tentar mudar para outro sistema operacional. Isto ocorre porque esta pessoa terá feito "investimentos duráveis em ativos complementares específicos" (Shapiro e Varian, 1999, p. 126) para este sistema. Ela já terá adquirido *softwares* específicos para este sistema que provavelmente não funcionarão no novo ambiente, além de ter que arcar (com seu tempo ou até financeiramente) com todo o treinamento necessário para aprender as funções do novo sistema operacional.

O aprisionamento tecnológico é um grande perigo para os consumidores, pois limita seu poder de barganha e os deixa vulnerável a comportamentos oportunistas por parte dos fornecedores. Para estes, porém, pode ser um fonte de lucros extraordinários, se bem administrado, além de servir como barreira à entrada de competidores potenciais em seu mercado. O Quadro 1 abaixo mostra os tipos possíveis de aprisionamento e seus custos de troca associados.

Quadro 1

Tipos de aprisionamento	***Custos de troca***
Compromissos contratuais	Indenizações compensatórias ou liqüidadas
Compras de bens duráveis	Substituição de equipamento; tende à cair a medida que o bem durável envelhece
Treinamento em marca específica	Aprender sobre um novo sistema, tanto custo direto quanto perda de produtividade; tende a aumentar com o tempo
Informação e bancos de dados	Conversão de dados para novo formato; tende a aumentar ao longo do tempo à medida que a coleção aumenta
Fornecedores especializados	Financiamento de novo fornecedor; pode aumentar com o tempo se as aptidões forem difíceis de encontrar / manter
Custos de busca	Custos combinados do comprador e do fornecedor; incluem o aprendizado sobre a qualidade de alternativas
Programas de lealdade	Quaisquer benefícios perdidos do fornecedor titular, mais a possível necessidade de reconstruir o uso cumulativo

Fonte: Shapiro e Varian, 1999, p. 140

Para tecnologias já estabelecidas, com uma grande base instalada, o uso de padrões proprietários permite às empresas um maior controle sobre seus clientes. Porém, novas tecnologias geralmente têm dificuldade em impor padrões fechados, devido à natureza de rede dos sistemas de informação. O uso de padrões abertos ou a formação de alianças com outros competidores para o uso de um padrão comum geralmente é a única alternativa que possibilita a entrada desta nova tecnologia no mercado.

I.4 - Economias de rede e feedback positivo

Enquanto a economia industrial tradicional é movida pela economia de escala, podemos dizer que a nova economia da informação é movida pela economia de rede. Os bens de informação, especialmente os *softwares*, formam redes virtuais entre seus proprietários ao serem utilizados: a rede de usuários de uma determinada planilha eletrônica, ou de um sistema operacional. As redes são caracterizadas pela possibilidade de conexão e compartilhamento de informação entre seus membros, como a troca de arquivos de texto entre usuários do Microsoft Word por exemplo. O valor, para o usuário, de uma rede estará relacionado diretamente com o seu tamanho: quanto maior a rede, mais pessoas poderão trocar informações com este usuário. O conceito de rede, porém, já era verificado em indústrias tradicionais, como as de comunicações ou transportes (o valor de uma rede ferroviária é tão maior quanto seu tamanho e número de interconexões), porém a diferença entre estas redes reais e as redes virtuais da economia da informação é que, nesta última, as ligações entre os nodos serão intangíveis (Shapiro e Varian, 1999). Os usuários de sistemas de informação de padrões comuns pertencem à mesma rede, pois seus bens de informação podem se comunicar e compartilhar recursos.

A utilidade para um usuário de determinado *software* de compartilhar arquivos com outras pessoas é função positiva da quantidade de pessoas que também usam o mesmo *software*, isto é, do tamanho da rede de usuários deste *software*. Isto significa que bens de informação apresentam externalidades positivas de rede (também chamadas de economias de escala do lado da demanda). Sempre que um bem de informação é adquirido por um consumidor, o seu valor (sua utilidade) irá aumentar para todos os consumidores. De maneira inversa, a cada usuário que pára de utilizar determinado bem ou serviço de informação, o valor do bem ou serviço diminui para os usuários restantes.

A presença de externalidades de rede positivas nos mercados de informação proporciona o aparecimento de *feedback* positivo nos mercados: à medida que um bem de informação ganha popularidade (ou seja, sua rede virtual aumenta), ele se tornará cada vez mais atraente para novos consumidores, e seu sucesso inicial irá gerar novos ganhos sucessivos em popularidade, em um círculo virtuoso auto-alimentado. Isto acontece pois, não havendo outras diferenças, os agentes preferem se ligar à uma rede grande do que à uma pequena, pois a primeira oferece maior utilidade para cada agente. Sendo assim, o bem ou serviço que despontar como o provável ganhador entre alternativas similares disputando o

mesmo *market share* irá inevitavelmente se tornar vencedor, numa espécie de profecia auto-realizável (Shapiro e Varian, 1999).

Isto ressalta a importância, por parte dos fornecedores, de convencer os consumidores de que o seu bem de informação, e não o da concorrência, será o produto dominante do mercado, o que geralmente é feito através de campanhas de *marketing*, por exemplo. As expectativas dos consumidores representam um papel fundamental para que um bem de informação alcance a massa crítica necessária para dar partida em um processo de crescimento da demanda puxada por *feedback* positivo.

Nem as economias de escala do lado da oferta, que os bens de informação apresentam devido à sua estrutura de custo peculiar, nem as economias de escala do lado da demanda (as externalidades de rede) são conceitos novos, ou observados apenas no mercado de informação. O que é novo, porém, é a combinação destes dois efeitos simultaneamente. Como ressaltam Varian e Shapiro (1999):

> *O resultado é um golpe duplo em que o crescimento do lado da demanda tanto reduz o custo do lado da oferta quanto torna o produto mais atraente para outros usuários – acelerando ainda mais o crescimento da demanda. A conseqüência é um feedback positivo extraordináriamente forte, que faz com que setores inteiros sejam criados ou destruídos bem mais rápido do que na era industrial. (p. 214)*

No caso específico do *software*, as externalidades positivas de rede são os fatores fundamentais para explicar o sucesso de um sistema ou aplicativo. O valor do sistema operacional Windows, da Microsoft, para seus clientes, não é dado tanto pelas suas características técnicas ou sua superioridade sobre os produtos concorrentes. Os clientes atribuem alto valor a ele devido a sua imensa base instalada, correspondente a quase 90% de todos os computadores pessoais existentes, constituindo o padrão de fato do setor. Isso proporciona grande poder à Microsoft, pois torna difícil para qualquer concorrente conseguir massa crítica suficiente para ameaçar sua fatia do mercado, e permite que a empresa usufrua de altíssimas margens de lucro (Shapiro e Varian, 1999).

I.5 – Rivalidade, excludabilidade e propriedade intelectual

Bens de informação são, por padrão, bens de consumo não-rival (Krugman, Wells e Myatt, 2005, p. 522). Isto é, o bem pode ser consumido por muitos agentes simultaneamente, sem prejuízo de nenhuma parte. O custo marginal de permitir acesso ao bem para um

consumidor adicional é zero.

Software, sendo um bem de informação, é então um bem não-rival. O uso de um software por um usuário não impede o uso de outros usuários. Porém a mesma característica dos bens de informação que permite a produção de unidades à custo ínfimo após a criação da primeira cópia faz com que estes bens sejam também não-excludentes por natureza. Isto é, a facilidade de cópia de um bem de informação, como um software, impede que se negue a disponibilidade do bem à certos indivíduos. Em outras palavras, a facilidade e baixo custo de cópia de um bem de informação implica, na prática, em disseminação indiscriminada do bem entre os consumidores.

Para impedir este cenário e permitir que os produtores de *softwares* (e de outros bens de informação), algo externo ao bem deve existir que permita aos ofertantes impedir o consumo do bem por partes indesejáveis, tornando-o um bem excludente. Isto é, na prática, restringir o consumo àqueles consumidores que tenham pago por suas cópias dos bens, bem como impedir que outros produtores copiem o bem e passem a vender suas próprias cópias (sem terem incorrido nos pesados custos fixos da criação da primeira cópia).

Este agente externo capaz de transformar os bens de informação em bens não-rivais excludentes são os direitos de propriedade intelectual, em geral na forma de leis de patentes e de *copyright*. Estes instrumentos legais concedidos pelo Estado ao criador do bem permitem que o mesmo possua um monopólio temporário sobre a exploração comercial do bem (Krugman, Wells e Myatt, 2005, p. 523). No caso específico de *softwares*, as proteções via *copyright* são mais comuns.

Capítulo II - Software livre

II.1 - Definição e características

Software livre ou de código aberto é, amplamente definido, todo *software* que proporcione ao usuário certas liberdades que não são encontradas no modelo de *software* proprietário, ou fechado, tradicional. São elas a liberdade: para executar o programa para qualquer fim, em qualquer ponto e a qualquer tempo; de estudar o funcionamento do programa e adaptá-lo às necessidades de quem o estuda; de redistribuição de cópias; e para melhorar o programa e publicar as melhorias (SOFTEX, 2005a). Além disso, é um modelo de desenvolvimento de *software* que explora a inteligência coletiva distribuída de participantes em comunidades, inclusive dispersas geograficamente, através da Internet (Weber, 2000).

O termo original em inglês para descrever este tipo de *software, "free software"*, se mostrou problemático com os anos, devido à ambigüidade da palavra *"free"* na língua inglesa: ela pode significar tanto "livre" como "gratuito". Esta confusão se tornou célebre com a explicação dada à mesma pelo próprio autor do tema, Richard Stallman: *"to understand the concept, you must think of free as in free speech, not as in free beer"* (Taurion, 2004, p. 17). Ou seja, "*free software"* é uma questão de liberdade, não de preço. Para contornar este problema, foi criado mais tarde o termo "*open source software"*, ou *software* de código aberto, em alusão ao fato de o código-fonte de um *software* livre ser aberto, ou seja, disponível para todos. Em português e em outras línguas, como espanhol e francês, esta ambigüidade não existe, pois usa-se palavras diferentes para representar a idéia de "*free as in free speech"* (livre) e "*free as in free beer"* (grátis). Assim, convencionou-se chamar esta categoria de software apenas como *software* livre, e, para simplificar a narrativa, é esta nomenclatura adotada neste trabalho, apesar de '*software* livre' e de 'código aberto' não serem sinônimos perfeitos, como será analisado adiante.

Software livre, em suma, se distingue do *software* convencional, chamado de *software* proprietário, por ter o seu código-fonte disponível para qualquer pessoa que o deseje. Em geral, ele é distribuído em forma de código-fonte, porém pode ser distribuído também em formato binário, desde que acompanhado do código-fonte ou que o mesmo seja facilmente adquirido, gratuitamente ou por uma quantia módica referente a custos de reprodução ou transmissão do mesmo, se houver.

Porém, mais importante do que o fato de o código-fonte do *software* livre ser aberto é o seu modelo de desenvolvimento. Tradicionalmente, um programa de computador é desenvolvido por grupos pequenos de programadores e engenheiros, confinado dentro dos limites de uma empresa que os emprega, em uma estrutura hierárquica e altamente centralizada. Raymond (2002) compara este modelo de desenvolvimento ao de uma construção de uma catedral, com alguns operários especializados construindo algo sob as instruções e diretivas de um arquiteto-chefe. O modelo de desenvolvimento do *software* livre, por sua vez, é comparado à um bazar, ou mercado. Milhares de pessoas operando ao mesmo tempo, sem nenhuma aparente ordem ou autoridade estipulada, resolvendo problemas em paralelo. Em tese, este modelo deveria falhar, porém na prática, era altamente eficiente.

Neste modelo de desenvolvimento, uma porção inicial do código-fonte de algum programa é liberado na Internet para os interessados usarem e aprimorarem. Usuários de todo o mundo passam então a usar o *software,* sugerir mudanças e desenvolver novas funcionalidades para o mesmo. Os códigos gerados por essas contribuições são submetidos de volta ao criador de software e são incorporados ao *software* original. Geralmente, o criador inicial do *software* é reconhecido como o líder do projeto, encarregado de selecionar o que deve ser incorporado ao *software* ou não e de resolver disputas.

Há ainda outro aspecto fundamental do *software* livre, que é o seu modelo de licenciamento. O código-fonte dos programas produzidos neste modelo de desenvolvimento, apesar de disponível para qualquer interessado, não está em domínio público. A razão disso é que os idealizadores do movimento não queriam que seus esforços fossem apropriados por pessoas e empresas que não dessem o devido crédito aos autores iniciais e, mais importante, não devolvessem à comunidade as melhorias que fizessem sobre o *software* livre original. Assim, foram desenvolvidas diversas licenças de uso para *software* livre, que permitissem que as liberdades iniciais proporcionadas pelo *software* fossem mantidas indefinidamente. A primeira licença deste tipo, e até hoje a mais utilizada em projetos de *software* livre, é a Licença Pública Geral GNU, escrita por Richard Stallman em 1991 (Free Software Foundation, 1991). Com esta licença, Stallman criou o conceito de "*copyleft*", que permite usar o direito à propriedade intelectual não para restringir o uso do software, como no modelo proprietário, mas para garantir que as liberdades que ele proporciona se perpetuem.

O software livre permite então que qualquer indivíduo adquira não só o software, mas também sua receita de funcionamento, à custos praticamente nulos. Dada a natureza

intangível do software, como toda informação digitalizada, o custo marginal de produção de uma nova cópia do mesmo é equivalente ao custo de reprodução de sua mídia, o que, no caso de um download da Internet, pode ser considerado zero. Somado à isso, toda inovação adicionada à um software livre é imediatamente disseminada por toda a comunidade, obrigatoriamente. Estas características dificultam o entendimento de como este modelo pode prosperar num ambiente capitalista e mais ainda, como pode concorrer com os software proprietários.

II.2 - Um breve histórico do software livre

O conceito de *software* com código-fonte disponível não é algo novo. Nas primeiras décadas após a invenção dos computadores, o *software* era visto como algo complementar ao *hardware*, isto é, aos computadores e *mainframes* da época, e era compartilhado livremente. Muitos destes primeiros computadores "se situavam em departamentos de ciência da computação de universidades [...] e em centros de pesquisa privados", onde o software "era tratado como ferramenta de pesquisa. A idéia de distribuir o código-fonte livremente era vista como algo natural da prática padrão de pesquisas" (Weber, 2000, p. 6, tradução livre[1]).

Assim, nas décadas de 60 e 70 do século passado, o compartilhamento de *softwares* e o uso de protocolos comuns e abertos de comunicação era a prática usual. Este desenvolvimento cooperativo de software era feito, no entanto, "de maneira altamente informal", e "sem que fossem feitos esforços para delinear direitos de propriedade ou para restringir o uso do *software* desenvolvido" (Lerner e Tirole, 2004, p. 5, tradução livre[2]).

Este cenário se mostrou problemático a partir da década de 1980, quando a AT&T "passou a exercer seus (supostos) direitos de propriedade relacionados ao *software* de sistema operacional UNIX, ao qual vários acadêmicos e pesquisadores privados de outras empresas haviam feitos contribuições" (*Id.*[3]).

Assim, em 1983, Richard Stallman, então um pesquisador do Laboratório de Inteligência Artificial do MIT, fundou a Free Software Foundation (FSF), uma fundação sem

1 "*... mainframe computers sitting in university computer science departments [...] and in corporate reserch facilities [...] were thought of and treated as tools for research. The idea of distibuting source code freely was seen as a natural offshoot of standard research parctice;*"

2 "*... undertaken on a highly informal basis. Typically no efforts to delineate property rights or to restrict reuse of the software were made*"

3 "*... began enforcing its (purpoted) intellectual property rights related to the operating system software UNIX, to which many academics and corporate researchers at other firms had made contributions*"

fins lucrativos criada para promover o desenvolvimento e disseminação de *software* livre de várias espécies. A FSF desenvolveu a primeira licença de uso para *software* livre, chamada de Licença Pública Geral (GPL, no original em inglês) e lançou o projeto GNU, que almejava desenvolver um sistema operacional completo, com todas as aplicações periféricas necessárias, que fosse totalmente compatível com o UNIX, porém livre, e licenciado sob a GPL. O nome GNU é um acrônimo recursivo para *"Gnu's Not Unix"*, simbolizando a compatibilidade do Sistema GNU com o UNIX, porém deixando claro suas diferenças filosóficas e legais. Esta licença acabou por ficar conhecida como a Licença Pública Geral GNU (GNU GPL), e é até hoje a licença mais usada por projetos de *software* livre em todo mundo. Considerando-se apenas os projetos listados no *site* SourceForge.net (o maior repositório mundial de projetos *software* livre), 69% dos mais de sessenta e cinco mil projetos desenvolvidos sob licenças livres utilizam a GNU GPL (SourceForge.net, 2005).

II.3 - Richard Stallman e o projeto GNU

Richard Stallman é visto como o grande idealizador do movimento do *software* livre, e a FSF como sendo fundamental por lançar as bases filosóficas, tecnológicas e legais do movimento (Revolution OS, 2001). Stallman foi o criador inicial de alguns dos mais importantes programas para UNIX, como o editor de texto Emacs, o GCC, um compilador livre para a linguagem C e o GDB, um depurador de código para o GCC (Weber, 2000).

O projeto GNU desejava ser um sistema completo, que substituísse o UNIX e todos os seus aplicativos por alternativas livres. Esta, porém, era uma tarefa grande, o os desenvolvedores foram primeiro desenvolvendo programas periféricos, que ainda rodavam sobre o núcleo do UNIX, para posteriormente desenvolverem o *kernel*, ou núcleo, do sistema (denominado *GNU Hurd*). Porém, o projeto escolheu inicialmente uma arquitetura extremamente avançada para o *kernel*[4] do sistema, o que tornava seu desenvolvimento complicado e demorado. Em 1991, já existiam alternativas livres para quase todos os aplicativos comuns em sistemas UNIX proprietários, exceto para o coração do sistema. Uma melhor descrição do que consiste um sistema operacional é dada por Linus Torvalds:

> *O principal sobre um Sistema Operacional é que você nunca deveria notar sua presença. Porque ninguém realmente usa um Sistema Operacional. Pessoas usam programas em seus computadores e a única missão na vida de um Sistema Operacional é ajudar estes programas a funcionarem. Então, um*

4 *Kernel, ou núcleo, é um termo usado para denominar a parte principal de um sistema operacional, responsável pela interface direta entre o hardware de um computador com os demais softwares.*

> *Sistema Operacional nunca faz nada sozinho. Ele está apenas esperando pelos programas pedirem por certos recursos ou pedirem por um certo arquivo no disco ou pedirem para serem conectados ao mundo externo. E então o Sistema Operacional aparece e tenta tornar mais fácil para as pessoas escreverem programas. (Revolution OS, 2001)*

Assim, as pessoas poderiam usar *softwares* livres em seus computadores, mas ainda necessitavam de um sistema operacional proprietário (o UNIX) para rodá-los. Este problema foi solucionado no fim de 1991 com o advento do Linux.

II.4 - Linus Torvalds e o Linux

O Linux era a peça que faltava para o funcionamento do sistema GNU. Linus Torvalds, então com 21 anos, era estudante de ciência da computação da Universidade de Helsinque, Finlândia, e criou um *kernel* rudimentar, baseado no UNIX, inicialmente como um *hobby*. Ele publicou sua criação, batizada de "Linux", como *software* livre licenciado sob a GNU GPL, em grupos de discussão na Internet, pedindo comentários e ajuda para melhorar o sistema, e obteve uma quantidade surpreendente de respostas (Weber, 2000). Este código-fonte inicial continha aproximadamente dez mil linhas de código apenas (Wikipedia, 2005).

O *kernel* Linux funcionava perfeitamente com os aplicativos do projeto GNU, tanto tecnicamente quanto também do ponto de vista ideológico e legal, já que ambos compartilhavam a mesma licença de uso. Os usuários apenas juntavam o Linux, o núcleo do sistema, com os aplicativos periféricos do projeto GNU (como editores de texto, compiladores e bibliotecas) para criar um sistema operacional totalmente livre: o sistema GNU/Linux.

A comunidade de desenvolvedores ao redor do Linux cresceu gradativamente nos dois anos seguintes, porém a partir de 1994, com o lançamento da primeira versão oficial do *kernel* Linux (a versão 1.0), o número de desenvolvedores e contribuições passou a crescer aceleradamente, com atualizações de código em bases semanais e até diárias (Weber, 2000). Isto pode ser explicado pelo surgimento de uma "*killer application"* para o sistema GNU/Linux, ou seja, uma utilidade, uma aplicação que o tornasse imprescindível. Um aplicativo que por si só incentivaria as pessoas a usarem o sistema GNU/Linux para poderem usufruir deste programa: o servidor *web* Apache.

O Apache consiste em um *software* que disponibiliza as páginas dos *sites* para visitantes na Internet. Seu desenvolvimento começou em 1993, baseado em um dos primeiros servidores *web* disponíveis sob domínio público, desenvolvido no National Center for Supercomputing

Applications (NCSA), nos EUA (Revolution OS, 2001). Ele possuía características muito superiores aos servidores *web* proprietários concorrentes da época, permitindo o desenvolvimento de *sites* mais complexos e seguros. Segundo Eric Raymond, "se você olhar na história do Linux, a curva de adoção do Linux e a curva de adoção da Internet coincidem exatamente" (Revolution OS, 2001). Agora então, os administradores de sistemas tinham uma razão bastante forte para usar o sistema operacional GNU/Linux.

Ao ulitizarem a combinação do servidor *web* Apache e o sistema operacional GNU/Linux, os administradores de rede podiam hospedar vários *sites* utilizando o mesmo hardware, algo que nenhum concorrente proprietário era capaz de fazer na época: para cada *site* hospedado era necessário um computador diferente (Revolution OS, 2001). Além disso, o protocolo de comunicação utilizado entre os *softwares* servidores e clientes de *sites* era (e ainda é) baseado num padrão de comunicações aberto, denominado protocolo HTTP (*HiperText Transfer Protocol*). Isso significava que todos os servidores do mercado seriam perfeitamente compatíveis com os navegadores dos usuários, sejam eles proprietários ou não. A oportunidade de aumentar instantaneamente a capacidade de hospedagem, a transparência da mudança para os usuários finais e o custo extremamente baixo da nova plataforma (resumindo-se a custos de treinamento de pessoal) provou ser uma combinação imbatível. Os benefícios superavam em grande parte os custos de troca inerentes à mudança tecnológica.

O servidor *web* Apache ganhou presença no mercado de servidores desde então, respondendo atualmente por praticamente 70% de todos os servidores *web* instalados no mundo (Netcraft, 2005). O *kernel* Linux atualmente é composto por quase seis milhões de linhas de código (Wikipedia, 2005).

Capítulo III – Produção de software livre

III.1 - Agentes econômicos produtores de software livre

Um projeto de software livre é tipicamente iniciado por apenas um desenvolvedor (ou no máximo um grupo muito pequeno de indivíduos), responsável pela ideia inicial do sistema e sua primeira implementação. Assim foi, por exemplo, com os softwares do projeto GNU, o sistema operacional Linux e o servidor de páginas web Apache, citados no capítulo anterior.

Tomada a decisão de liberar seu projeto como software livre, costuma caber ao desenvolvedor inicial a tarefa de coordenação dos outros desenvolvedores em torno do sistema, principalmente no que se refere à decisões sobre novas funcionalidades, resolução de conflitos e manutenção da qualidade do código-fonte gerado.

Alternativamente, um indivíduo ou uma empresa pode resolver tornar livre um sistema seu já existente, até então proprietário, buscando os benefícios provenientes do modelo de desenvolvimento livre, apontados no capítulo anterior. A tarefa de liderança sobre o projeto ainda é necessária, e geralmente é capitaneada pelo agente detentor do software original ou algum agente externo nomeado por ele para tal, como uma organização ou fundação criada para este propósito.

Caso o projeto tenha sucesso em atrair o interesse de outros usuários e desenvolvedores além do grupo inicial de indivíduos, a comunidade formada em torno do projeto tende a ser bastante heterogênea, formada por usuários (que não contribuem com código-fonte) e desenvolvedores (que contribuem com código-fonte) (Weber, 2000). Pode haver também empresas interessadas no software produzido pelo projeto, e estas também podem ser divididas entre usuárias e desenvolvedoras (quando seus funcionários são diretamente alocados à contribuição para o projeto).

Não há nenhum mistério quanto aos motivos que levam usuários e empresas a fazerem uso de projetos de software livre. Basta aos agentes examinarem as funcionalidades providas pelo software livre em questão frente às suas demandas técnicas e econômicas, comparando-o inclusive com alternativas proprietárias. No entanto, é curioso, pelo menos a princípio, o comportamento de indivíduos e empresas que contribuem ativamente para projetos de software livre. O que motiva, por exemplo, programadores talentosos a escolherem

voluntariamente alocar parte de seu tempo e de sua capacidade mental (ambos recursos escassos e valiosos) para criação de algo que é, essencialmente, um bem público, e cujo trabalho não será diretamente compensado, na maioria dos casos?

Outra questão importante é entender o que leva um empresa competitiva a alocar seus recursos (especificamente, o tempo de seus funcionários) para contribuir com projetos de software livre, produzindo bens que ela não controla totalmente e que estarão à disposição, pela própria natureza do software livre, de todos os seus competidores.

Para responder estas questões, devemos analisar os diferentes tipos de incentivos que influem nas decisões dos indivíduos e das empresas quando estes decidem alocar seus recursos à produção de software livre. Faremos isso primeiramente analisando o comportamento dos indivíduos, sejam eles programadores agindo por conta própria ou seguindo orientação de seu empregador. Em seguida, será analisado o comportamento das empresas produtoras de software livre, de acordo com diferentes modelos de negócio tipicamente empregados por estas empresas.

III.2 – Motivações dos indivíduos

III.2.1 – Porque um indivíduo contribui para um projeto de software livre?

Segundo Lerner e Tirole (2000), "um programador participa de um projeto, seja comercial ou de código aberto, apenas se ele deriva um benefício líquido (amplamente definido) ao ingressar na atividade. O benefício líquido é igual ao *payoff* imediato (benefício atual menos custo atual) mais o *payoff* futuro." (p. 14, tradução livre[5]). No caso de um projeto de software proprietário tradicional, o principal benefício imediato derivado pelo programador é uma recompensa monetária: seu salário enquanto empregado de uma empresa competitiva ou os lucros de sua própria empresa, obtidos da exploração direta da propriedade do software sendo produzido.

Tal benefício porém é tipicamente inexistente quando o programador participa de um projeto de software livre. Assim, o aspecto surpreendente no desenvolvimento de software livre é o fato de que indivíduos estão dispostos a investir e cooperar mesmo na ausência de pagamento direto e invocações de direito de propriedade do produto de seu trabalho (Kogut e

5 *"A programmer participates in a project, wheter commercial or open source, only if she derives a net benefit (broadly defined) from engaging in the activity. The net benefit is equal to the immediate payoff (current benefit minus current cost) plus the delayed payoff."*

Metiu, 2001). Outros incentivos devem então se fazer presentes para explicar o comportamento do programador participante de projetos de software livre.

Estes incentivos são diversos e com graus de impacto variados. Como observa Rossi (2004), o universo de projetos de software livre existentes é extremamente heterogêneo, com os projetos variando entre si de maneira significativa quanto à complexidade, propósito, modularidade, intensidade de contribuições, entre outros fatores. Além disso, dentro de um único projeto, tipicamente os "participantes se diferenciam em razão da intensidade de suas contribuições, motivação primária para contribuir, nível de comprometimento ideológico, etc" (p. 2, tradução livre[6]). Assim, é improvável que apenas um incentivo poderá servir para explicar a motivação de um grupo tão vasto e heterogêneo de indivíduos.

A literatura pesquisada então se propõe a examinar diversos tipos possíveis de incentivos identificados nos projetos de software livre existentes, bem como incentivos identificados em outras indústrias com características similares. Kogut e Metiu (2001), Rossi (2004) e Lerner e Tirole (2000) analisam as motivações dos indivíduos separando-as em dois grandes grupos: as motivações extrínsecas, pertencentes ao ramo mais tradicional da teoria econômica; e as motivações intrínsecas, derivadas da satisfação por efetuar uma tarefa em si.

III.2.2 – Incentivos extrínsecos

Estando a recompensa pecuniária, a mais comum forma de motivação extrínseca, fora do âmbito de um projeto de software livre tradicional, há que se achar outros motivos que levem os programadores a contribuírem seu tempo escasso à um ou mais projetos de software livre. Lerner e Tirole (2000 e 2004) foram um dos primeiros economistas a investigarem esta questão. Estes autores identificaram uma outra forte motivação de caráter extrínseco agindo sobre os programadores de software-livre, a qual chamaram de incentivo de sinalização (*signaling incentive*, no original).

Este incentivo de sinalização age sobre os benefícios futuros que o programador obtém de seu trabalho. Os benefícios imediatos derivados de seu trabalho não são diferentes do trabalho de um programador trabalhando em um software proprietário: o conserto de algum problema (*bug*) no sistema, a existência de uma nova funcionalidade necessária para si mesmo ou seu grupo, etc. A única diferença se dá na ausência de recompensa monetária para

6 *"participants differ as regards the intensity of their contributions, the primary motivation for contributing, the level of ideological commitment etc."*

o programador de software livre, em geral.

O benefício futuro se refere a incentivos relacionados à carreira do programador e à gratificação de seu ego. Um bom trabalho realizado hoje pode ser significativo para acesso a futuras ofertas de trabalho, participação em empresas comerciais ou acesso ao mercado de capital de risco (Lerner e Tirole, 2000, p. 14). A gratificação do ego do programador provém do desejo de reconhecimento de sua qualidade e potencial por seus pares.

Os autores reconhecem que os programadores em geral respondem a ambos os incentivos (carreira e ego), porém em graus variados de acordo com cada indivíduo. De qualquer maneira, dada a semelhança destes incentivos do ponto de vista econômico, agrupam-os sob o nome de incentivo de sinalização.

Apesar do incentivo de sinalização ser aplicável a todos os programadores, de software-livre ou não, ele é mais impactante em projetos livres, dada a natureza de acesso direto ao código-fonte produzido pelos programadores. (Holmstrom, 1999) estipula que incentivos de sinalização são mais fortes onde a performance é mais visível para a audiência relevante; o impacto da performance é maior; e a performance contém mais informação sobre a capacidade ou talento do agente. Assim, quando se trata de software livre, a "abertura do código fonte permite melhor medida da performance e aumenta a visibilidade da contribuição de um indivíduo para a audiência relevante"[7]. Observadores externos podem identificar não apenas a contribuição de cada indivíduo, mas a dificuldade de cada tarefa, como o problema foi abordado e, em muitos casos, todos os passos tomados pelo programador até a eventual implementação final da solução, graças à sistemas de versionamento de código-fonte disponíveis. Além disso, como cada programador é o único responsável por suas contribuições individuais, "o código fonte disponível para inspeção (…) reflete mais diretamente seu talento[8]" (Rossi, 2004, p.3, tradução livre). O incentivo de sinalização é então mais influente em projetos de software livre por este tipo de desenvolvimento de software proporcionar mensuração da performance individual de cada programador mais abrangente e mais precisa.

O incentivo de sinalização porém não é capaz de explicar por completo o comportamento dos indivíduos que produzem software livre. Rossi (2004) argumenta que esta

7 *"the openness of the source code allows better performance measurement and increases the visibility of one's own contribution to the relevant audience."*

8 *"the source code available to anyone for inspection more directly reflects her talent;"*

teoria não explica porque um programador já de "alto nível" decidiria iniciar um projeto livre em primeiro lugar, fato este comum e observado no mundo real. O próprio Linus Torvalds, criador inicial do sistema operacional Linux cita o desejo de participar em uma comunidade de desenvolvedores com os quais ele se identificava (uma motivação intrínseca) como motivação primordial para a liberação do código-fonte da primeira versão do Linux sob uma licença livre (Torvalds, 1998).

Outra crítica à dominância do incentivo de sinalização sobre a motivação dos programadores é feita por Weber (2000), que aponta que, fosse este incentivo (que para este autor é chamado de reputação) a principal força por trás da motivação dos programadores, haveria um número muito maior do que o observado na prática de desafios à autoridade dos líderes dos projetos livres e mais desvios (chamados tecnicamente de *forks*) dos sistemas desenvolvidos. Isto se dá pois a capacidade de um indivíduo receber crédito pelo uma contribuição específica fica totalmente à mercê do(s) líder(er) do projeto: é ele que decidirá, em última instância, se aquela contribuição é necessária e boa o suficiente para fazer parte do sistema sendo desenvolvido. Caso seja decidido o contrário, o trabalho do desenvolvedor terá sido em vão. Neste caso, o programador rejeitado poderia sentir o desejo de criar sua própria versão separada do projeto (seu *fork*) e aplicar sua contribuição e atuar como o líder do seu ramo. Em seguida, tentaria convencer outros desenvolvedores a migrar para a sua versão do sistema. Na prática, porém, este comportamento (conhecido como *forking*) é um evento muito raro, pois "há forte pressão social contra forking de projetos. Isso não acontece exceto sob um apelo de forte necessidade, com muita justificação pública e com uma renomeação [do projeto]"[9] (Raymond, 1999, apud Rossi, 2004, p. 4, tradução livre).

Por último, o incentivo de sinalização não explica por que um grande número de desenvolvedores escolhe investir seu tempo escasso em tarefas menos "gloriosas" no desenvolvimento de software livre, como criação de comentários, documentação, material de suporte, testes, divulgação, etc.. Se um indivíduo planeja maximizar a exibição de seu talento para atingir uma maior reputação entre seus pares e possíveis contratantes, por que ele escolheria efetuar tarefas que pouco ou nada demonstrem suas habilidades? No entanto, muitos indivíduos assim o fazem na prática nos projetos de software livre (Rossi, 2004, p. 4).

Outro tipo de incentivo presente nos projetos de software livre e analisado na literatura

9 *"there is strong social pressure against forking projects. It does not happen expect under a plea of dire necessity, with much public justification and with a renaming."*

pesquisada é o incentivo extrínseco que advém quando o programador cria alguma funcionalidade ou conserta algum *bug* cujo impacto é diretamente relevante para ele. Imagine um programador que até então é apenas usuário de um software livre particular e gostaria que o sistema possuísse uma funcionalidade qualquer que atualmente não possui. Ou então ele encontra um defeito no sistema que atrapalha seu caso de uso específico. Confrontado com este cenário, o programador pode resolver desenvolver a funcionalidade extra que deseja, ou consertar o *bug* que lhe atrapalha. Dado o caráter viral da maioria das licenças de uso de software livre, o programador seria obrigado à devolver ao projeto suas melhorias. Porém o mesmo pode ocorrer com licenças mais permissivas. Como diz Rossi (2004):

> *Necessidades dos usuários constituem um poderoso incentivo para criar o código do software em primeiro lugar e, dado o baixo custo associado ao ato de tornar o código disponível via um meio tão poderoso como a rede, a decisão de distribuí-lo de graça não parece tão surpreendente. (p. 5, tradução livre)* [10]

Inovações no produto a partir de seus usuários, com compartilhamento das mesmas à fabricantes e outros usuários é um fenômeno já observado em outros campos, segundo Rossi (2004). A disseminação gratuita da inovação irá ocorrer, de maneira geral, quando os benefícios da mesma superem os custos. Como já vimos, o custo de difusão da inovação gerada pelo usuário do software livre (e agora desenvolvedor também) é mantido baixo devido à difusão se dar através da Internet, via de regra.

O custo de oportunidade do desenvolvedor em aprender o básico necessário para compreender o sistema para modificá-lo pode ser baixo também devido ao chamado efeito "*alumni*" (Lerner e Tirole, 2000). Devido à natureza em geral gratuita do software livre, e o fato de seu código-fonte ser disponível, é comum que o primeiro contato dos usuários e desenvolvedores com software livre se dê nas escolas e universidades, onde tais softwares costumam ser amplamente empregados. O programador já terá absorvido então o custo de estudo inicial do sistema durante sua formação acadêmica.

Outro fator levantado por Rossi (2004) que contribui para a diminuição do custo de oportunidade do programador prestes a liberar sua inovação como software livre provém do perfil heterogêneo dos indivíduos da comunidade. Muitos deles são ainda estudantes que "tipicamente não consideram potenciais adotantes de sua inovação como rivais" (p. 5,

10 *"User needs indeed constitute a powerful incentive to create the software code in the first place and, given the low costs associated to the act of making the code available through a means as powerful as the web, the decision to distribute it for free may not appear particularly startling."*

tradução livre) [11].

Estabelecido um baixo custo associado à difusão gratuita da inovação gerada pelo usuário a partir de sua necessidade inicial, resta então analisar os possíveis benefícios provenientes da decisão de compartilhar tal inovação de maneira livre. Um benefício direto é a externalidade positiva de rede classicamente associada à bens de informação, como software. O valor do software para cada usuário aumenta conforme mais pessoas fazem uso do software. Ao liberar sua contribuição de volta ao projeto, o usuário-desenvolvedor garante que todos os outros usuários do projeto passarão a usar sua inovação, aumentando a rede de usuários da mesma e conferindo uma maior utilidade para todos o seus usuários.

Outro benefício mais indireto provém do fato que, ao contribuir sua inovação como software livre, o desenvolvedor poderá contar com mais pessoas para melhorar ainda mais seu código, seja consertando problemas e *bugs*, seja criando novas funcionalidades além das originalmente idealizadas. O programador terá acesso à um *pool* de recursos (o tempo e capacidade intelectual dos demais programadores da comunidade) que ele nunca teria caso mantivesse sua inovação guardada para si, à um custo efetivamente zero.

Essa contribuição à projetos de software livre a partir das necessidades dos usuários serve para explicar a contribuição à partes menos "gloriosas" (que possibilitam menor ganho de reputação) dos projetos. Ao ajudar na documentação, teste, divulgação dos projetos de software livre, os indivíduos estão ativamente preenchendo suas necessidades de aprendizado no uso ou desenvolvimento do sistema, ou ajudando a aumentar o valor do software para si ao aumentar a quantidade de usuários para o mesmo.

Há uma consequência interessante desse incentivo à contribuição: os projetos de software livre, apesar de serem um bem público não-excludente, costumam tolerar bem e até se beneficiarem da presença de um certo número de *free-riders* na comunidade, isto é, indivíduos que usam o sistema sem contribuir nada em retorno, desde que haja um núcleo de contribuintes forte e ativo. Isto se dá pelo fato de que um número maior de usuários aumenta a possibilidade do descobrimento de *bugs* no sistema, que poderão ser reportados pelos *free-riders* eventualmente, mesmo que apenas por frustração (Weber, 2000). Uma vez identificado o *bug*, ele pode ser consertado pelo núcleo ativo do projeto, melhorando o sistema para todos. Assim sendo, assumindo que a comunidade em torno de um projeto é bastante heterogênea

11 *"... a significant fraction of the contributors to F/OSS projects is made up by students who typically do not regard potential adopters of their innovation as rivals."*

em função das diferentes capacidades e disponibilidades de seus indivíduos, bem como no perfil de uso e participação no sistema, ao aumentar o número de usuários do sistema, a probabilidade de contribuições ao código do sistema tende à aumentar. Isto é uma característica particular do software livre, oposta à visão tradicional de bens públicos provenientes de ações coletivas, onde tipicamente

> *(...) grandes grupos são menos prováveis de gerar bens coletivos devido à dificuldade de que indivíduos têm em fazer suas contribuições particulares visíveis, e devido aos problemas de coordenação entre grandes números [de indivíduos]. (Weber, 2000, p. 29, tradução livre)*[12]

Estes incentivos extrínsecos analisados baseados em reputação e melhora de seu trabalho a partir de *feedback* e contribuição de seus pares constituem algo muito similar ao processo tradicional de pesquisa científica acadêmica. Algo muito natural considerando que as origens do movimento do software livre, como visto no capítulo anterior, se deram nas grandes instituições de pesquisas, quando do surgimento dos primeiros computadores no ambiente acadêmico. O movimento do software livre transfere então os ideais de pesquisa científica para a produção de software: o compartilhamento de informações, possibilitando a melhora dos projetos através de *feedback* e avaliação por pares; o ganho de prestígio e reconhecimento pelo trabalho efetuado; e o avanço do conhecimento coletivo. Todos estes aspectos estão presentes no ambiente da pesquisa acadêmica e nos projetos de software livre (Bonaccorsi e Rossi, 2003).

Em diversas pesquisas empíricas (Bonnacorsi e Rossi 2003, Hawkins 2002, Lakhani e Wolf 2005, Raymond 2002, Revolution OS 2001), porém, parte significante dos desenvolvedores entrevistados citam outras motivações além dos incentivos já reportados acima como os principais motivos que os levam à contribuir com software livre. Estes incentivos advém do próprio ato da programação ou da participação em algo maior do que eles, e podem ser agrupados na chancela de incentivos intrínsecos.

III.2.3 – Incentivos intrínsecos

Motivações intrínsecas são definidas por Ryan e Deci (2000) como:

> *... a execução de uma atividade por sua satisfação inerente, ao invés de alguma consequência separada. Quando intrinsecamente motivada, uma*

12 *"(...) where large groups are less likely to generate collective goods because of the difficulties that individuals have in making their particular contributions visible, and because of coordination problems among large numbers."*

> *pessoa é movida a agir pela diversão ou desafio implicados, e não devido a estímulos, pressões ou recompensas externas.*[13] *(apud Rossi, 2004, p. 2)*

A necessidade humana por competência e autonomia estão diretamente ligadas aos sentimentos de interesse e satisfação em uma tarefa (Lakhani e Wolf, 2005), e são a base para as teorias dos incentivos intrínsecos, que em geral são separados em dois grupos: os incentivos baseados na satisfação intrínseca ao executar uma tarefa; e os incentivos baseados em um compromisso para com a comunidade, ou baseados em princípios (Lakhani e Wolf, 2005). No primeiro grupo se enquadram ações como por exemplo aprender a tocar um instrumento musical nos fins de semana, sem nenhum intuito de obter qualquer retorno via esta ação a não ser a própria satisfação inerente a tocar o instrumento, dominar gradualmente seu uso e conquistar o desafio apresentado. No segundo grupo, está a satisfação proveniente de atuar de acordo com as normas e valores esperados de uma comunidade a qual o indivíduo se identifica, como por exemplo seguir os valores morais da sociedade ao fazer uma "boa ação", ou ao doar tempo ou dinheiro para uma instituição de caridade.

A satisfação obtida pela execução de uma tarefa é maximizada, segundo Lakhani e Wolf (2005), quando a habilidade do indivíduo se equipara à dificuldade da tarefa. Uma tarefa que esteja além da capacidade da pessoa trará sentimentos de ansiedade. Já uma tarefa simples demais proporcionará tédio ao executor. A natureza informal do ambiente em torno do software livre, associado ao fato de que um programador poderá sempre escolher o que deseja programar (afinal, ele é plenamente responsável pelos seus atos enquanto contribuidor de um projeto livre) significa que o software livre proporciona um potencial de maximização da satisfação maior do que projetos proprietários tradicionais, onde o programador deve trabalhar em resposta às demandas das funcionalidades determinadas pela empresa produtora do software (Rossi, 2004, p. 7).

O prazer inerente à programação de algo que se tenha interesse é um dos principais motivos citados por programadores de software livre como razão para contribuir com um projeto (Lakhani e Wolf, 2005). Em especial, pode-se destacar o senso de criatividade relacionado ao cumprimento das tarefas. Programadores obtém satisfação maior ao resolver um problema heurístico (sem um caminho pré-determinado para a solução, não algorítmico) de maneira inovadora e apropriada (Amabile 1996, apud Lakhani e Wolf, 2005, p. 5).

13 *"... the doing of an activity for its inherent satisfactions rather than for some separable consequence. When intrinsically motivated, a person is moved to act for the fun or challenge entailed rather than because of external prods, pressures or rewards."*

Outros motivos comumente citados por participantes em projetos de software livre (Raymond 2002, Torvalds 1998, Lakhani e Wolf 2005) dizem respeito à comprometimento com ideologias como as dos fundadores do movimento de software livre (acreditam que todo software deve ser livre, que isto seria uma questão de ética); ou à identificação com a cultura *hacker,* ou com a comunidade formada em torno de um ou mais projetos. Estes incentivos intrínsecos são caracterizados no grupo de incentivos relacionados à princípios, e "têm sido geralmente descritos como decorrentes de uma cultura de reciprocidade generalizada, de doações, ou de um claramente definido senso de identificação com a comunidade de software livre"[14] (Rossi, 2004, p. 7, tradução livre).

A associação entre comunidades de software livre com comunidades regidas por uma economia de abundância é tema recorrente na literatura pesquisada. Nestas comunidades, o status social é determinado "não por aquilo que você controla, mas por aquilo que você doa" (Raymond, 1999, apud Rossi, 2004, p. 8). A frequente doação de código-fonte entre os membros da comunidade criaria a obrigação social da reciprocidade e estabeleceria laços de interdependência social entre os indivíduos. Esta visão porém, é também bastante contestada. Rossi (2004) argumenta que o papel das licenças de uso de software livre, com seus caráter viral, é fundamental para garantir parte da reciprocidade das doações dos indivíduos, ao garantir que o código contribuído não poderá ser apropriado unilateralmente por apenas um agente ou uma parte da comunidade. Weber (2000) contesta o caráter de abundância sugerido, pois apesar de a quantidade de capacidade intelectual das comunidades como um todo parecer um estoque infinito de recursos em prol do desenvolvimento dos projetos, o tempo e foco mental individual de cada programador não é. Cada indivíduo da comunidade em torno de um projeto deverá maximizar alguma função de utilidade para determinar a sua participação ou não no projeto, de que maneira isso será feito e em que quantidade.

Ainda assim, a importância do sentido de identificação com a comunidade é apontado como um poderoso incentivo para a participação individual em projetos de software livre em diversos estudos empíricos até então (Rossi, 2004), seguidos pelo sentimento de realização própria e satisfação com as tarefas executadas (Rossi 2004 e Lakhani e Wolf 2005).

14 *"Obligation-based intrinsic motivations have been variously described as arising from a generalized reciprocity, gift-giving culture, or a clearly defined sense of identification with the F/OSS community that may or may not entail the sharing of an explicit anti-Microsoft attitude."*

III.3 – Motivações das empresas

As firmas não estão sujeitas à nenhum incentivo intrínseco à sua participação em projetos de software livre, obviamente, posto que a satisfação em atuar em tais tarefas é algo obtenível apenas por seres humanos. Isto significa que as motivações atuantes sobre uma empresa competitiva quando da decisão de atuar ativamente em algum projeto livre serão exclusivamente extrínsecas, isto é, baseadas em expectativas de benefício futuro em consequência da participação no projeto em questão.

De maneira resumida, podemos dizer que as empresas competitivas, apesar de não poderem se apropriar direta e exclusivamente de suas contribuições à um software livre, procuram se beneficiar indiretamente através de produtos ou serviços proprietários complementares à este software livre, que não são providos de maneira eficiente pelo projeto de software livre em questão (Lerner e Tirole, 2000). Quando analisadas em maior detalhe, porém, as motivações são tão variadas quanto os modelos de negócio das empresas e suas diferentes relações com projetos de software livre.

No modelo onde a empresa possui um produto ou serviço complementar à um projeto de software livre, batizado de "convivência simbiótica" por Lerner e Tirole (2004, p. 12), a empresa terá grandes incentivos à adquirir *know-how* extenso sobre o projeto livre, e a melhor maneira de fazer isso é alocando parte de seus recursos à participação (em diferentes níveis) no projeto. Exemplos de empresas neste modelo são a Red Hat e Canonical, que fornecem "distrubuições linux" e serviços de consultoria e suporte associados. Estas distribuições são constituídas de milhares de softwares livres amplamente disponíveis, porém empacotados de maneira à criar um bem único de maior valor (Shapiro e Varian, 2004).

Uma empresa usuária de um software livre já existente, seja como um produto de uso interno em sua cadeia produtiva ou administrativa ou mesmo como parte integrante do produto ou serviço proprietário por ela fornecido, terá algumas motivações à participar ativamente do desenvolvimento deste software livre. Primeiramente, sua participação "implica em um crescimento da base de usuários e consequentemente aumento da popularidade do projeto, com os usuais benefícios em termos de efeitos de redes"[15] (Rossi, 2004, p. 21, tradução livre) examinados no primeiro capítulo: o projeto torna-se mais valioso para a empresa e aumenta consideravelmente a chance de manutenção do sistema ao longo do

15 *"... it implies an enlargement of the user base and consequently an increase in popularity of the project, with the usual benefits in terms of network effects."*

tempo por outros indivíduos e empresas além dela própria.

Outro benefício decorrente da participação ativa da empresa neste projeto livre é a possibilidade de influenciar ou até direcionar o desenvolvimento do software, em termos de novas funcionalidades por exemplo, na direção que for mais conveniente para a empresa, ou pelo menos evitar desvios prejudiciais à si. Obviamente, o sucesso deste benefício dependerá do grau de influência da empresa e seus empregados sobre os outros membros do projeto, que por sua vez tende a variar positivamente de acordo com a quantidade e intensidade da participação histórica da empresa no projeto (Raymond, 2000).

Lerner e Tirole (2004, p.12) identificam outros modelos de negócios de empresas competitivas relacionadas ao software livre. Uma empresa cujo produto compita diretamente com um ou mais projetos de software livre pode temporariamente encorajar seus funcionários à engajarem nestes projetos, de maneira a avaliar as qualidades e deficiências de seus competidores. Alem disso, as empresas podem participar de projetos livres visando geração de boas relações públicas com programadores externos e consumidores.

Há ainda a possibilidade da empresa criar um novo projeto de software livre (e necessária estrutura de governança sobre o mesmo) a partir de código-fonte previamente existente e até então proprietário. Segundo Lerner e Tirole (2004), isto ocorre quando a empresa opera no modelo de convivência simbiótica, em uma estratégia similiar a prover o barbeador gratuitamente (o código-fonte liberado) para vender mais lâminas (o produto ou serviço complementar que a empresa espera vender) (p. 13). A empresa colocará esta estratégia em prática caso estime que a lucratividade alcançada com o aumento das vendas do produto ou serviço complementar será maior do que o lucro que seria gerado caso o código-fonte inicial não houvesse sido liberado.

Outro fator ser considerado quando se analisa as motivações para empresas atuarem ativamente nas comunidades de software livre, segundo Rossi (2004), é o tamanho das empresas. Grandes firmas tendem a relatar motivações diferentes de pequenas e médias empresas em pesquisas empíricas (Wichman, 2002 e Bonaccorsi e Rossi, 2003, apud Rossi, 2004, p.21).

Nas empresas menores, natureza das motivações que levam à contribuição com projetos livres tende a ser similar aos incentivos extrínsecos de programadores individuais.

Particularmente, necessidades específicas das empresas que levaram à criação de novos softwares internos devido à ausência de alternativas aceitáveis (tanto livres ou proprietárias) tendem a ser o principal incentivo à criação de novos softwares. Uma vez criados, o custo de liberar estes softwares para à comunidade tende à ser baixo nestas pequenas empresas, pois em geral as mesmas incorreriam e altos custos de transação ao tentar vender software de uso tão específico. Aliado à aos custos baixos, o potencial para aproveitamento de externalidades positivas de rede e os benefícios sociais na imagem da empresa são a receita para efetuar a liberação deste código-fonte em formato livre.

Já as empresas maiores, ainda segundo Rossi (2004, p. 21), tendem a serem motivadas mais pelo uso de software livre como componentes de baixo custo em seus produtos e serviços proprietários; pelo aumento de compatibilidade entre seus produtos e projetos de software livre existente; pelo desejo de adoção de padrões livres não controláveis pela sua concorrência; ou ainda movidos por decisões estratégicas, como aumentar a visibilidade e funcionalidades de um projeto de software livre afim de ameaçar softwares proprietários de empresas comerciais concorrentes.

Capítulo IV – Estudo de caso: micro empresa desenvolvedora de software

A fim de verificar na prática a presença (ou não) dos diversos incentivos à produção de software livre, este autor analisou uma empresa produtora de software livre. A empresa em questão foi escolhida devido ao conhecimento prévio do autor da empresa e de seus sócios, o que garantiu amplo acesso aos dados da empresa, e pelo fato da empresa e seus sócios já haverem participado ativamente em projetos de software livre.

Para este estudo foi entrevistado um dos sócios da empresa, que também atua como programador, Otávio Sampaio. A entrevista buscou compreender o tamanho da empresa; o perfil de uso de software livre na empresa; os perfis dos projetos de software livre que a empresa (ou seus sócios e funcionários) contribuiu; e as razões que os moveram à tais participações.

IV.1 – A empresa estudada

A Riopro Informática Ltda é uma micro empresa produtora de softwares voltados para gestão empresarial. A empresa produz tanto software sob demanda, customizado para as necessidades específicas do cliente, quanto software "empacotado", não personalizado. O foco atual da empresa está justamente neste segundo modelo de software, pois o mesmo permite obter maiores economias de escala e consequentemente, maior lucro.

A empresa não comercializa nenhum software livre, porém todos os seus produtos atuais possuem no mínimo algum componente livre. Além disso, software livre é utilizado para desenvolver os produtos e gerenciar o dia a dia da empresa, tal qual enviar e receber e-mails, sistemas de segurança de rede, planilhas, etc.

A empresa está no mercado há 8 anos, possui 3 sócios e 3 funcionários e faturamento na faixa de 1 a 2 milhões de reais por ano.

IV.2 – Uso de software livre

O uso de software livre é amplamente disseminado na empresa. 90% dos computadores do parque computacional da empresa (incluindo aí os *desktops* dos sócios e funcionários e os servidores da empresa) rodam o sistema operacional Linux, na distribuição Ubuntu. Os outros 10% usam sistema operacional proprietário Windows, porém apenas por razões de

compatibilidade com sistemas legados da empresa e de clientes, bem como necessidade de interfacear com sistemas externos à empresa que requeiram o uso de Windows (como alguns serviços de órgãos públicos, por exemplo).

Além dos sistemas operacionais, o uso de programas de software livre nas tarefas cotidianas também à alto, mesmo nos computadores que rodam Windows: o navegador, os programas de planilhas, edição de textos e apresentação, são todos software livre.

Segundo o entrevistado, as razões para o uso de software livre ao invés de alternativas proprietárias são, em ordem decrescente de importância: segurança (as alternativas proprietárias costumam ser de pior qualidade e mais susceptíveis à falhas de segurança); funcionalidades exclusivas (os softwares livres utilizados possuem funcionalidades não encontradas em similares proprietários); funcionalidades superiores (os softwares livres utilizados executam tarefas com mais competência); maior liberdade de uso e customização; amplo suporte disponível (no caso, através da comunidade de usuários e desenvolvedores de software livre); baixo custo; compatibilidade ideológica.

O uso de software livre também é amplo nos produtos e serviços desenvolvidos e fornecidos pela empresa. Segundo o entrevistado, absolutamente todos os softwares desenvolvidos atualmente pela empresa são desenvolvido com o uso extensivo de ferramentas livres: sistemas de controle de versão (ex: Git, Subversion), ambientes de desenvolvimento integrado (IDE) (ex: Netbeans, RubyMine), ferramentas de testes automatizados (ex: RSpec, Webrat), bancos de dados (ex: MySQL, PostgreSQL, MongoDB) além das próprias linguagens de programação (ex: Ruby, Java). As razões dadas para o uso destas ferramentas são as mesmas, e na mesma ordem de importância, das apontadas acima para o uso de softwares livre de emprego mais amplo.

Além do uso das ferramentas, há também amplo uso de componentes de software livre (bibliotecas, *frameworks*, etc) que são integrados aos produtos da empresa. Estes componentes são usados com intuito de ganhar tempo de desenvolvimento, pois assim e empresa não precisa "re-inventar a roda" ao programar partes comuns dos sistema já implementadas de forma genérica via software livre (tais como *drivers* de acesso a bancos de dados e mapeamento de dados, subsisteamas de criptografia, autenticação e autorização de usuários, bibliotecas de leitura e conversão de formatos de arquivos, etc). Outro fator importante, segundo o entrevistado, é o acesso a componentes cujo código-fonte engloba

conhecimentos não dominados pela equipe de desenvolvedores da empresa, permitindo acessar recursos em um nível de qualidade que a empresa não possuiria de outra maneira, à um custo extremamente baixo.

IV.3 - Produção de software livre

Apesar da empresa não produzir software livre como um produto final, ela o faz como um subproduto de seus sistemas. Segundo o entrevistado, aproximadamente uma dezena de projetos de software livre já foram iniciados e disponibilizados pela empresa ou por seus sócios e funcionários como consequência direta do desenvolvimento dos produtos e serviços da empresa.

Isso geralmente ocorre quando uma parte do sistema sendo desenvolvido é extraído em um subcomponente de uso amplo o suficiente para interessar a outros desenvolvedores. Quando se identifica esta oportunidade, a empresa considera se o código-fonte a ser extraído e disponibilizado poderá interessar a uma quantidade mínima de outros desenvolvedores (que o entrevistado não soube precisar um número exato, porém notou que bastam alguns poucos interessados para dar continuidade ao projeto). Caso a estimativa geral da empresa seja de que o componente será interessante para outrem, procedem à criação de um novo projeto de software livre.

Este processo de liberação de um componente como software livre possui um custo de oportunidade atrelado que não é ignorado pela empresa. Para que o projeto a ser criado tenha a mínima chance de sucesso, isto é, de atrair outros desenvolvedores interessados, ele precisa estar bem organizado e documentado. Há que se criar no mínimo uma página na internet para ser a "casa" virtual do projeto. Alguém da empresa há que ficar responsável pelo projeto, sua liderança, e pelo contato e coordenação com outros desenvolvedores interessados. Além disso, muitas vezes o código-fonte a ser liberado precisa de pequenas (ou nem tão pequenas) alterações antes de ser disponibilizado, pois é comum a necessidade de remover partes que não se deseja compartilhar, melhorar os comentários embutidos no código, aumentar a quantidade de testes unitários ou ainda traduzir a documentação para o inglês (idioma padrão da indústria) visando maior alcance do projeto.

O mais recente caso deste tipo se deu com um subsistema de geração de relatórios em arquivos PDF a partir de bases de dados XML, via web, que estava sendo usado pelo principal software da empresa, um sistema de controle de ativos patrimoniais.

Mas qual o objetivo de fornecer toda essa propriedade intelectual a potenciais competidores? O entrevistado citou duas principais motivações igualmente poderosas, em sua opinião: a vontade de devolver à comunidade de software livre algo útil, visto que a empresa usufrui tanto do que a comunidade lhe fornece; e a possibilidade de receber de volta melhorias (na forma de novas funcionalidades e correções de problemas) nos sistemas liberados como software livre. Foram citados também outras razões, porém tidas como de menor importância pelo entrevistado: o desejo de participar ativamente em comunidades específicas de certos softwares; desejo de aumentar a visibilidade e reputação da empresa e de seus sócios e funcionários; criar um portfólio de fácil acesso que demonstre a capacidade técnica da empresa e de seus membros; gerar menções sobre a empresa (marketing) em certos círculos, visando atrair bons funcionários, parceiros ou clientes.

Além de contribuir com software propriamente dito, a empresa estimula seus sócios e funcionários a participarem ativamente na comunidade de software livre, principalmente nas comunidades relacionadas aos softwares que a empresa usa e depende no dia e dia e em seus produtos. Essa participação se dá através de publicações de dicas, tutoriais e documentação em geral sobre uso dos softwares livres no blog da empresa, em listas e fóruns de discussão e em palestras e eventos em todo país.

Neste caso, as motivações citadas pela empresa são, novamente, o desejo de retribuir à comunidade e angariar reputação entre os pares, além do fato de, ao disseminar este conhecimento, a empresa acredita que também está aprendendo novas coisas, tanto conhecimento específico sobre os softwares livres em questão quanto, no mínimo, melhorando a capacidade de relacionamento inter-pessoal de seus sócios e funcionários.

Pode-se perceber nas motivações indicadas pelo entrevistado participação igualmente importante de motivações intrínsecas e extrínsecas quando da decisão em produzir software livre, apesar de um maior número de motivações citadas serem de caráter intrínsecas. Além disso, as razões apresentadas são amplamente consistentes com o verificado na literatura pesquisada, particularmente a busca por benefícios extrínsecos provenientes das externalidades positivas de rede do software livre.

CONCLUSÃO

A participação de indivíduos e empresas na produção de software livre se apresenta inicialmente como um dilema, pois frente à ausência de recompensa pecuniária ao esforço dos agentes, somos obrigados à buscar outras motivações não triviais para explicar este comportamento. Comportamento este amplamente verificado na indústria há pelo menos 15 anos, data de surgimento e / ou amadurecimento dos principais projetos de software livre existentes, como o Linux e o servidor *web* Apache.

No entanto, as teorias econômicas existentes sobre os incentivos extrínsecos e (principalmente) intrínsecos dos agentes, bem como o entendimento do software enquanto bem de informação (sujeito à externalidades positivas de rede e baixo custo de reprodução) nos permite desmistificar o assunto. Indivíduos respondem à incentivos extrínsecos que lhe permitem obter benefícios futuros devido à contribuição com projetos livres. Além disso, a satisfação inerente à resolução de problemas complexos de maneira heurística, aliado aos baixos custos de oportunidade propiciados pelas características idiossincráticas do *software* faz com que os incentivos intrínsecos também possuam papel de suma importância nas decisões dos indivíduos ao contribuir com projetos livres.

Já as empresas competitivas apresentam diversos incentivos extrínsecos à participar de projetos de software livre, porém atrelados ao modelo de negócios específico da empresa, bem como o perfil de relacionamento com os softwares livres em questão. No caso mais comum, é estabelecido uma convivência simbiótica entre a empresa e o software livre, onde o produto da empresa é algo complementar ao software livre.

Fica claro, também, que a heterogeneidade dos agentes participantes em um projeto de software livre é determinante para o sucesso do mesmo, pois permite que potencialmente todos os diferentes tipos de incentivos atuem ao mesmo tempo, em membros diferentes do projeto. Isto significa que é a captação dos talentos mais talentosos e motivados, e não a motivação média dos membros do projeto, que será mais importante para o sucesso do software livre.

O estudo de caso apresentado no capítulo IV permitiu que se observasse na prática a

influência dos principais incentivos extrínsecos e intrínsecos mencionados anteriormente. Do ponto de vista da empresa, o potencial acesso à mão de obra capacitada para os projetos internos posteriormente liberados se destacou como o principal motivador para a produção de software livre, visando melhorias e correções para sua base de código-fonte.

Os incentivos intrínsecos, por sua vez, se destacaram como motivador dos sócios e funcionários da empresa estudada. O desejo de retribuir à comunidade a qual os indivíduos se identificavam, efetivando uma reciprocidade esperada pelos membros de tal comunidade, foi o principal vetor de influência na decisão de atuar em projetos de software livre dos indivíduos pesquisados.

REFERÊNCIAS BIBLIOGRÁFICAS

BONACCORSI, A.; ROSSI, C. **Contributing to the common pool resources in Open Source software. A comparison between individuals and firms.** Sant'Anna School of Advanced Studies, Institute for Informatics and Telematics (IIT-CNR), Pisa, 2005. 37p.

BONACCORSI, A.; ROSSI, C. Why open source software can succeed. **Research Policy**, v. 32, n. 7, p. 1243-1258, julho 2003.

BROWNE, C. **The Economics of free software**. Disponível em: <http://linuxfinances.info/info/freeecon.html>. Acesso em: 15 de abril de 2005.

FREE SOFTWARE FOUNDATION. **GNU General public license**. 1991. Disponível em: <http://www.gnu.org/copyleft/gpl.html>. Acesso em: 15 de abril de 2009.

FREEH, V; MADEY, G; TYNAN, R. The open source software development phenomenon: an analysis based on social network theory. In: AMERICAS CONFERENCE ON INFORMATION SYSTEMS, 8, 2002. **Proceedings**... Dallas: AMCIS, 2002. p. 1806-1813.

GALLAUGHER, J. M.; WANG, Y. Network effects and the impact of free goods: an analysis of the web server market. **International journal of electronic commerce**, Saddle River, v. 3, n. 4, p. 67-88, 1999.

GARZARELLI, G. **Open source software and the economics of organization**. Dipartimento di Teoria Economica e Metodi Quantitativi per le Scelte Politiche, Università degli Studi di Roma, Roma, 2002. 21 p. Trabalho não publicado.

GUROVITZ, H. **Linux: O fenômeno do software livre**. São Paulo: Abril, 2002. 89 p. (Super Interessante: Coleção Para saber mais)

HAWKINS, R. E. **The Economics of Open Source Software for a Competitive Firm: Why give it away for free?** Pennsylvania State University, Dubois, 2002. 18p.

HEALY, K; SCHUSSMAN, A. **The ecology of open-source software development**. University of Arizona, 2003. 24 p. Trabalho não publicado.

HOLMSTROM, B. Managerial Incentive Problems: A Dynamic Perspective. **Review of Economic Studies**, Massachusetts , v. 66, n. 1, p. 169-182, 1999.

KOGUT, B.; METIU, A. Open source software development and distributed innovation. **Oxford review of economic policy**, Oxford, v.17, n. 2, p. 248-264, jun. 2001.

KRUGMAN, P.; WELLS, R.; MYATT, A. **Microeconomics: Canadian Edition**. Nova York: Worth Publishers, 2005. Primeira edição.

KSHETRI, N. Economics of Linux adoption on developing countries. **IEEE Software**, Washington, v. 21, n. 1, p. 74-81, jan./fev. 2004.

LAKHANI, K. R.; WOLF, R. G. **Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects.** Perspectives on free and open source software, The MIT Press, Massachusetts, 2005. p. 36-54.

LERNER, J; TIROLE, J. **The simple economics of open source.** NBER Working Papers Series, National Bureau of Economic Research, Cambridge. Working Paper 7600, março 2000. Disponível em

<http://www.nber.org/papers/w7600>.

LERNER, J; TIROLE, J. The economics of technology sharing: open source and beyond. **Harvard NOM Research Paper**, Boston, v. 4, n. 35, p. 1-43, novembro 2004.

LERNER, J; TIROLE, J. **The scope of open source licensing**. Harvard University, 2002. 53 p. Trabalho não publicado. Disponível em <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.465&rep=rep1&type=pdf>.

LIEBOWITZ, S. J.; MARGOLIS, S. E. **Network Externalities (Effects).** 2005. Disponível em: <http://www.utdallas.edu/~liebowit/palgrave/network.html>. Acessado em: 17 de julho 2008.

NETCRAFT. **July 2005 Web Server Survey**. 2005. Disponível em: <http://news.netcraft.com/archives/2005/07/01/july_2005_web_server_survey.html> Acessado em: 28 de julho de 2005.

OPEN SOURCE INITIATIVE. **The Open source definition**. 1997. Disponível em: <http://www.opensource.org/docs/definition_plain.html>. Acesso em: 15 de maio de 2007.

PINK, D. H. **Drive: the surprising truth about what motivates us.** Nova York: Riverhead Books, 2009.

POLZIN, P. **A Indústria brasileira de software e a distribuição eletrônica de software**. Recife: DEPE-UFPE, Prêmio Brazilian Electronic Journal of Economics, 1998.

RAYMOND, E. S. **The Cathedral and the bazaar**: musings on Linux and open source by an accidental revolutionary. 2002. Disponível em: <http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/>. Acesso em: 15 de abril de 2005.

RAYMOND, E. S. **The Magic cauldron**. 2000. Disponível em: <http://www.catb.org/~esr/writings/magic-cauldron/magic-cauldron.html>. Acesso em: 15 de abril de 2005.

REVOLUTION OS. Produção e direção de J. T. S. Moore. EUA, Wonderview Productions, LLC, 2001. 1 DVD (85 min.): NTSC, son., color. Legendado. Documentário.

RIEHLE, D. **The Economic Motivation of Open Source Software: Stakeholder Perspectives.** IT Professional, Abril 2007. IEEE Computer Society. p. 25-32.

ROSSI, C.; BONACCORSI, A. **Intrinsic motivations and profit-oriented firms in Open Source software. Do firms practise what they preach?** Sant'Anna School of Advanced Studies & University of Pisa, Pisa, 2005. 32p. Trabalho não publicado. Disponível em <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.145.5401&rep=rep1&type=pdf>.

ROSSI, C.; BONACCORSI, A. **Why profit-oriented companies enter the OS field? Intrinsic vs. extrinsic incentives.** Sant'Anna School of Advanced Studies, Pisa, 2005. 5p.

ROSSI, M. A. **Decoding the "free/open source (F/OSS) puzzle"** - a survey of theoretical and empirical contributions. Siena: Dipartimeno di economia politica, Università degli studi di Siena, 2004. 42 p.

SAMPAIO, O. F. C. **A Indústria de software**: situação atual e perspectivas. 2001. 48 f.. Dissertação (Graduação em Economia). Instituto de Economia, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2001.

SCHMIDT, K. M.; SCHNITZER, M. **Public subsidies for open source?** Some economic policy issues of the software market. University of Munich, 2002. 37 p. Trabalho não publicado. Disponível em <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.4325&rep=rep1&type=pdf>.

SHAPIRO, C; VARIAN, H. **A Economia da informação**: como os princípios econômicos se aplicam à era da Internet. Rio de Janeiro: Campus, 1999.

SHAPIRO, C; VARIAN, H. **Linux adoption in the public sector**: an economic analysis. University of California, 2004. 26 p. Trabalho não publicado. Disponível em <http://people.ischool.berkeley.edu/~hal/Papers/2004/linux-adoption-in-the-public-sector.pdf>.

SOFTEX. **A Indústria de software no Brasil - 2002**: fortalecendo a economia do conhecimento. Campinas: Softex, 2002. 80 p.

SOFTEX. **O Impacto do software livre e de código aberto na indústria de software do Brasil**. Campinas: Softex, 2005a. 80 p.

SOURCEFORGE.NET, **Software map by license**, 2005. Disponível em: <http://sourceforge.net/softwaremap/trove_list.php?form_cat=13>. Acessado em: 29 de julho de 2005.

SPILLER, D; WICHMANN, T. **Basics of open source software markets and business models**. Berlim: Berlecon Research, 2002. 58 p.

SPOLSKY, J. **Strategy letter V**. 2002. Disponível em: <http://www.joelonsoftware.com/articles/StrategyLetterV.html>. Acesso em: 15 de abril de 2005.

TAURION, C. **Software livre**: potencialidades e modelos de negócio. Rio de Janeiro: Brasport, 2004.

TORVALDS, L. What motivates free developers? **First Monday**, v. 3, n. 3, Março, 1998. Disponível em <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/583/504>. Acessado em 15 de Maio de 2008.

VARNER, P. E. **The Economics of open source software**. 1999. Disponível em: <http://www.cs.virginia.edu/~pev5b/writing/econ_oss/index.html>. Acesso em: 15 de abril de 2005.

VEPSTAS, L. **Is free software inevitable?** 2001. Disponível em: <http://www.linas.org/theory/freetrade.html>. Acesso em: 07 de maio de 2005.

WEBER, S. **The political economy of open source software**. University of Berkley, California, 2000. 41 p. Trabalho não publicado. Disponível em <http://brie.berkeley.edu/publications/wp140.pdf>.

WHEELER, D. **A. More than a gigabuck**: estimating GNU/Linux's size. 2002. Disponível em: <http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html>. Acesso em: 07 de maio de 2005.

WHEELER, D. A. **Why open source software / free software (OSS/FS, FLOSS, or FOSS)? Look at the numbers!** 2005. Disponível em: <http://www.dwheeler.com/oss_fs_why.html>. Acesso em: 15 de abril de 2005.

WIKIPEDIA, **Linux kernel**, 2005. Disponível em: <http://en.wikipedia.org/wiki/Linux_kernel>. Acessado em: 29 de julho de 2005.

WILLIAMS, S. **Free as in freedom**: Richard Stallman's crusade for free software. 2002. Disponível em: <http://www.oreilly.com/openbook/freedom/index.html>. Acesso em: 15 de abril de 2005.